Maneuver Classification for Aircraft Fault Detection

Nikunj C. Oza, Irem Y. Tumer, Kagan Tumer, and Edward M. Huff
Computational Sciences Division
NASA Ames Research Center
Mail Stop 269-2
Moffett Field, CA 94035-1000
{oza,itumer,kagan,huff}@email.arc.nasa.gov

May 8, 2003

Abstract

Automated fault detection is an increasingly important problem in aircraft maintenance and operation. Standard methods of fault detection assume the availability of either data produced during all possible faulty operation modes or a clearly-defined means to determine whether the data provide a reasonable match to known examples of proper operation. In the domain of fault detection in aircraft, identifying all possible faulty and proper operating modes is clearly impossible. We envision a system for online fault detection in aircraft, one part of which is a classifier that predicts the maneuver being performed by the aircraft as a function of vibration data and other available data. To develop such a system, we use flight data collected under a controlled test environment, subject to many sources of variability. We explain where our classifier fits into the envisioned fault detection system as well as experiments showing the promise of this classification subsystem.

1 Introduction

A critical aspect of the operation and maintenance of aircraft is detecting problems in their operation when they occur in flight. This allows maintenance and flight crews to fix problems before they become severe and lead to significant aircraft damage or even a crash. Fault detection systems designed for this purpose are becoming a standard requirement in most aircraft [4, 10]. However, most systems produce too many false alarms, mainly due to an inability to compare real behavior with modeled behavior, making their reliability questionable in practice [9]. Other systems require a clearly-defined means to determine

Table 1: Conceptual open loop model illustrating assumed causal relationships.

Flight $\longrightarrow$ Maneuver (M)	Aircraft $\longrightarrow$ Attitude (A)	Physical $\longrightarrow$ Input (I)	Internal $\longrightarrow$ Response (R)	Measured Output (O)
•Fwd. Flight •Side Flight •Fwd. Climb •Fwd. Descent •Hover •Hover Turn •Coord. Turn •*[other]*	•Radar Alt. •Airspeed •Climb Rate •Heading •Bank •Pitch •Side Slip •*[other]*	•Engine Torque •Engine Speed •*[Mast Lifting]* •*[Mast Bending]* •*[other]*	•*[Tooth Bending]* •*[Backlash]* •*[Friction]* •*[Heat]* •*[other]* •*[DAMAGE]*	•Vibration - x axis - y axis - z axis •*[Temp]* •*[Noise]* •*[other]*

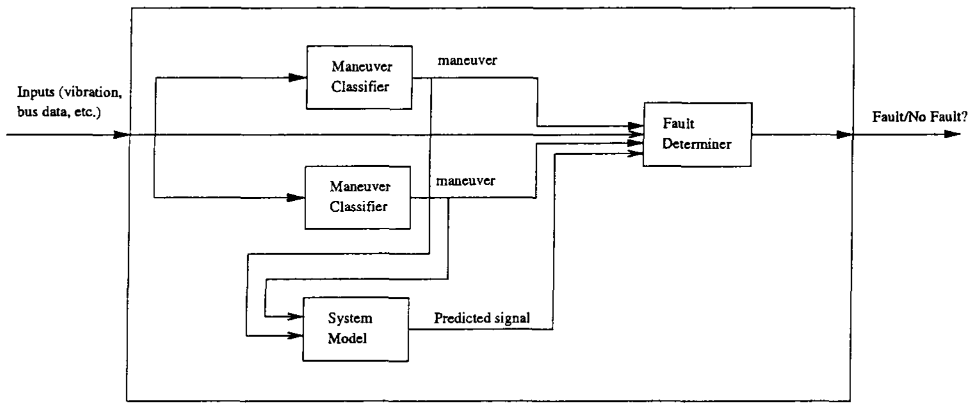

Figure 1: Online Fault Detection System Block Diagram.

whether the data provide a reasonable match to known examples of proper operation or assume the availability of data produced during all possible faulty operation modes [4, 5, 10]. Because of the highly safety-critical nature of the aircraft domain application, most fault detection systems must function well even though fault data are not available and the set of possible faults is unknown. Models are typically used to predict the effect of damage and failures on otherwise healthy (baseline) data [6, 9]. However, while models are a necessary first start, the modeled system response often does not take operational variability into account, resulting in high false-alarm rates. Novelty detection is one approach to overcoming this problem, addressing the problem of modeling the proper operation of a system and detecting when its operation deviates significantly from normal operation [5, 7].

In this paper, we present an approach to novelty detection for in-flight aircraft data. The data were collected as part of a research effort to understand the sources of variability present in the actual flight environment, with the purpose of reducing the high rates of false alarms [6, 12]. In past work, we have described aircraft operation conceptually according to the open-loop causal model shown in table 1. We assume that the maneuver being performed (M) influences the observable aircraft attitudes (A), which in turn influence the set of possibly observable physical inputs (I) to the transmission. The physical inputs influence the transmission in a variety of ways that are not typically observable (R). However, there are outputs that can be observed (O). Our approach to fault detection in aircraft depends fundamentally on the assumption that the nature of the relationships between the elements M, A, I, R, and O described above change when a fault materializes. Many approaches to fault detection attempt to model only the set of possible outputs (O) and indicate the presence of a fault when the actual outputs do not match the model. However, this approach is difficult because the output space is often too complicated to allow faithful modeling and measuring differences between the modeled and actual outputs. This latter difficulty remains even if one attempts to model the output as a function of something that influences it such as the physical inputs or the flight maneuver. Approaches to fault diagnosis (e.g., [17]) attempt to predict either normal operation or one of a designated set of faults. As stated earlier, this is not possible in the aircraft domain because the set of possible faults is unknown and fault data are non-existent. For this reason, we envision a fault detection system containing a classifier that models the flight maneuver (M) as a function of the outputs (O). This allows us to measure differences between modeled and actual operation in the space of flight maneuvers, which is a much simpler space than the space of vibration signals (O). We would like to harness this fact in our system.

Figure 1 is a block diagram of the system that we envision for online fault detection. A fundamental idea is the use of multiple sources of information to predict aspects of the state of the system being modeled, such as the maneuver being performed and predicting faults when the system state predictions are incompatible. In this paper, we present several maneuver classifiers, which are depicted in figure 1 in the top two blocks marked "Maneuver Classifier." These classifiers take vibration data from various accelerometers and/or other available data as input and predict the maneuver being performed. Multiple classifiers that predict the maneuver may be present in the fault detection system. Models of aircraft operation that generate predictions of vibration signatures, which are in the mold of traditional fault detection systems that we described earlier, may also be included in this system (the lowest box marked "System Model"). The goal of this work is to develop a fault detection system which compares the maneuver predictions from the various maneuver classifiers and uses other appropriate data to diagnose whether a fault is present based on these predictions. For example, if a vibration data-based classifier predicts that the helicopter is flying forward at high speed, but other data and/or subsystems indicate that the aircraft is on the ground, then the probability that a fault is present is high. Additionally, our fault detection system can use physical constraints. For example, if the predicted maneuver fluctuates more rapidly than what is physically

Table 2: Flight Protocol for Each Phase of Experiment.

		Obs. Order	Ground & Hover		Primary Flight Maneuvers						Hover & Ground	
Pilot 1	Flight 1	1	G	H	A	B	C	D	E	F		
		2			B	C	D	E	F	A		
		3			C	D	E	F	A	B	H	G
	Flight 2	1	G	H	I	J	K	L	M	N		
		2			J	K	L	M	N	I		
		3			K	L	M	N	I	J	H	G
Pilot 2	Flight 3	1	G	H	D	E	F	A	B	C		
		2			E	F	A	B	C	D		
		3			F	A	B	C	D	E	H	G
	Flight 4	1	G	H	L	M	N	I	J	K		
		2			M	N	I	J	K	L		
		3			N	I	J	K	L	M	H	G

possible, then we hypothesize that the probability that a fault is present is high.

In the remainder of this paper, section 2 discusses the two aircraft studied and the data generated from them. We discuss the machine learning methods that we used and the data preparation that we performed in order to use these methods in section 3. We discuss the experimental results in section 4. We summarize the results of this paper and discuss ongoing and future work in section 5.

2 Aircraft Data

The data used in this work were collected from two helicopters: an AH1 Cobra and OH58c Kiowa [6]. The data were collected by having two pilots each fly two designated sequences of steady-state maneuvers according to a predetermined test matrix (table 2). It uses a modified Latin-square design to counterbalance changes in wind conditions, ambient temperature, and fuel depletion. Each of the four flights consisted of an initial period on the ground with the helicopter blades at flat pitch, followed by a low hover, a sequence of maneuvers drawn from the 12 primary maneuvers, a low hover, and finally a return to ground (the list of maneuvers is shown in table 3). Each maneuver was scheduled to last 34 seconds in order to allow a sufficient number of cycles of the main rotor and planetary gear assembly to apply the signal decomposition techniques used in the previous studies [6].

Summary matrices were created from the raw data by averaging the data produced during each revolution of the planetary gear. The summarized data consists of 31475 revolutions of data for the AH1 and 34144 revolutions of data

Table 3: Aircraft Maneuvers for Phases 1 and 2.

Maneuver	Name	Symbol	Description
A	Forward Flight, Low Speed	FFLS	Fly straight, level, & forward at 20 kts.
B	Forward Flight, High Speed	FFHS	Fly straight, level, & forward at 60 kts.
C	Sideward Flight Left	SL	Fly straight, level, & sideward left.
D	Sideward Flight Right	SR	Fly straight, level, & sideward right.
E	Forward Climb, Low Power	FCLP	Fly forward, straight, & climb at 40 psi.
F	Forward Descent, Low Power	FDLP	Fly forward, straight, & descend at 10 psi.
G	Flat Pitch on Ground	G	Vehicle on ground skids.
H	Hover	H	Stationary hover.
I	Hover Turn Left	HTL	Level hover, turning left.
J	Hover Turn Right	HTR	Level hover, turning right.
K	Coordinated Turn Left	CTL	Fly level, forward, & turning left.
L	Coordinated Turn Right	CTR	Fly level, forward, & turning right.
M	Forward Climb, High Power	FCHP	Fly forward, straight, & climb at 50 psi.
N	Forward Descent, High Power	FDHP	Fly forward, straight, & descend at 50 psi.

for the OH58c. Each row, representing one revolution, indicates the maneuver being performed during that revolution as well as the following 30 quantities: Revolutions per minute of the planetary gear, torque (mean, standard deviation, skew, and kurtosis), and vibration data from six accelerometers (root-mean-square, skew, kurtosis, and a binary variable indicating whether signal clipping occurred). For the AH1, the mean and standard deviations were available for the following attitude data from a 1553 bus: altitude, speed, rate of climb, heading, bank angle, pitch, and slip.

3 Methods

Sample torque and RPM data from one maneuver separated by pilot and by flights are shown in figures 2 and 3, respectively. The highly-variable nature of the data, as well as differences due to different pilots and different days when the aircraft were flown, are clearly visible and make this a challenging classification problem. To perform the necessary mapping for this problem, we chose multilayer perceptrons (MLPs) with one hidden layer and radial basis function (RBF) networks as base classifiers. Furthermore, we constructed ensembles of each type of classifier, as well as ensembles consisting of half MLPs and half RBF networks, because ensembles have been shown to improve upon the performance of their constituent or base classifiers, particularly when the correlations among them can be kept low [2, 18]. We now explain these methods in more detail.

3.1 Multilayer Perceptrons

The multilayer perceptron (MLP) is the most common neural network representation (see [1] for a more detailed explanation). It is often depicted as a

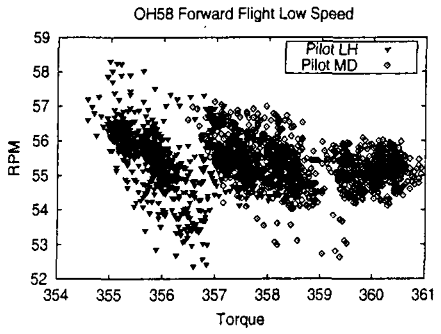

Figure 2: OH58c Maneuver A (Forward Flight Low Speed) Pilot-Separated

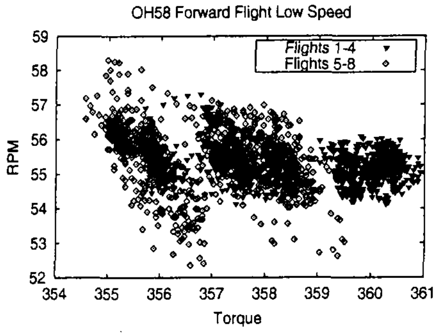

Figure 3: OH58c Maneuver A (Forward Flight Low Speed) Flight-Separated

directed graph consisting of nodes and arcs—an example is shown in figure 4. Each column of nodes is a *layer*. The leftmost layer is the *input layer*. The inputs of an example to be classified are entered here. The second and third layers are the *hidden layer* and the *output layer*, respectively. Information flows from the input layer to the hidden layer and then to the output layer via a set of arcs. The nodes within a layer are not connected to each other. In our example, every node in one layer is connected to every node in the next layer, but this is not required in general. Also, an MLP can have more or less than one hidden layer and can have any number of nodes in each hidden layer. One hidden layer is considered standard for applications in order to avoid overfitting (fitting the training data so precisely that the network captures artifacts specific to that training set and therefore performs poorly on new data). The number of hidden units is normally chosen in the manner we did, which is by training with different numbers of hidden units and choosing the number that yields the highest performance on a separate dataset not used for training.

Each non-input node, its incoming arcs, and its single outgoing arc constitute a *neuron*, which is the basic computational element of an MLP. Each incoming arc multiplies the value coming from its origin node by the weight assigned to that arc and sends the result to the destination node. The destination node adds the values presented to it by all the incoming arcs, transforms it with a nonlinear *activation* function (e.g., a sigmoid function), and then sends the result along the outgoing arc. For example, the value returned by a hidden node z_j in our example MLP is

$$z_j = g\left(\sum_{i=1}^{A} w_{i,j}^{(1)} x_i\right)$$

where A is the number of input nodes, $w_{i,j}^{(k)}$ is the weight on the arc in the kth layer of arcs that goes from unit i in the kth layer of nodes to unit j in the next

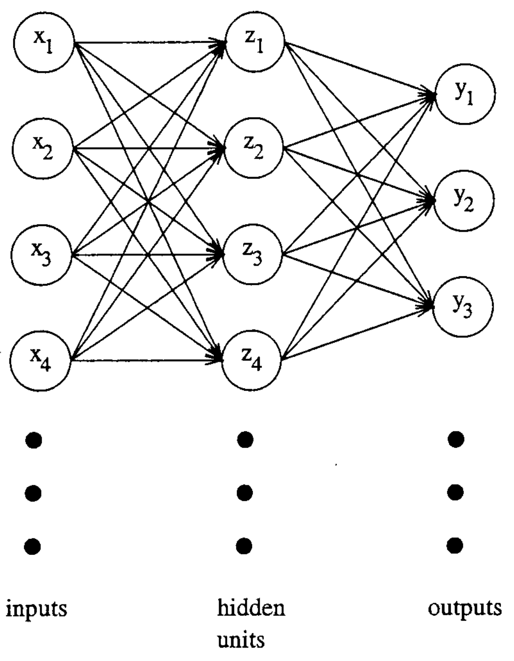

Figure 4: An example of a feedforward multilayer perceptron.

layer (so $w_{i,j}^{(1)}$ is the weight on the arc that goes from input unit i to hidden unit j), g is a nonlinear activation function, and $|A|$ is the number of input nodes. A commonly used activation function is the sigmoid function:

$$g(a) \equiv \frac{1}{1 + exp(-a)}.$$

The value returned by an output node y_j is

$$y_j = g\left(\sum_{i=1}^{Z} w_{i,j}^{(2)} z_i\right)$$

where Z is the number of hidden units. The outputs are clearly nonlinear functions of the inputs. MLPs used for classification problems typically have one output per class. The example MLP depicted in figure 4 is of this type. The outputs lie in the range $[0, 1]$. Each output value is a measure of the network's confidence that the example presented to it is a member of that output's corresponding class. Therefore, the class corresponding to the highest output value is returned as the prediction.

MLP learning performs nonlinear regression given a training set. The most widely used method for setting the weights in an MLP is the *backpropagation* algorithm [3, 11]. For each example in the training set, its inputs are presented to the input layer of the network and the predicted outputs are calculated. Then the difference between each predicted output and the corresponding target output is calculated. The gradient of this error with respect to the weights on the network's arcs is also calculated. The weights are then adjusted according to this gradient so that if the training example is presented to the network again, then the error would be less. The learning algorithm typically cycles through the training set many times—each cycle is called an *epoch* in the neural network literature.

For both data sets, we determined the number of hidden units experimentally. For MLPs, we explored hidden layer sizes ranging from 5 to 100 in increments of 5, and settled on 25 hidden units for the AH1 and 65 units for the OH58c. We used a learning rate and momentum term of 0.2, and trained for 100 epochs. The performances of the MLPs were fairly insensitive to the number of hidden units and kernels and the learning parameters. We created 280 MLPs for each helicopter—each MLP was given a different set of random initial weights before training, but were trained using the same training sets.

Past work [8] has asserted that MLPs are not as well-suited to fault detection as distance-based classifiers because real data often falls outside the range of the training data, requiring the MLP to extrapolate, which it does not do as well as distance-based classifiers. The truth is not so simple. In particular, in order to use a distance-based classifier, one has to choose a distance function that properly weighs all the attributes relative to each other. Choosing such a distance function is very difficult. Also, the approach in [8] uses a nearest-neighbor classifier to classify examples as coming from normal operation or one of a designated set of faulty operation modes. As discussed earlier, this approach is not applicable to our aircraft fault detection domain because the set of possible faults is unknown and fault data is nonexistent. Also, as asserted in [8], nearest-neighbor classifiers do not require significant training time because training consists of merely storing the training examples. This is clearly impractical because of the amount of training data collected.

3.2 Radial Basis Function Networks

As its name implies, radial basis function (RBF) networks (see [1] for a more detailed explanation) create a set of spherical basis functions from the training set, and then fit the outputs using these basis functions. In particular, the fitted function is of the form

$$y_k(x) = \sum_{j=1}^{J} w_{k,j}\phi_j(x)$$

where the J basis functions $\phi_j(x)$ ($j \in \{1, 2, \ldots, J\}$) are

$$\phi_j(x) = exp\left(-\frac{||x - \mu_j||^2}{2\sigma_j^2}\right),$$

and k indexes over the outputs—just as with MLPs, RBF networks used for classification typically have as many outputs as possible classes. The functions $\phi_j(x)$ are Gaussian radial basis functions with means μ_j and variances σ_j^2. These parameters are determined in the first stage of RBF network learning using an algorithm that finds the means and widths of clusters of data points in the input space (the space of x values). Therefore, the outputs (y's) are not used in the first stage of learning. The cluster means and widths correspond to means and variances of the radial basis functions. The second stage determines the parameters $w_{k,j}$. These parameters represent different possible output values. The predicted output for a new input is calculated as a linear combination of the $w_{k,j}$'s weighted by $\phi_j(x)$, which measures how far the new example is from the center of the j'th RBF.

For the RBF networks, we used 100 centers for the OH58c data and determined each kernel's center and width using the nearest 300 patterns.[1] For the AH1 data, we used 55 kernels with the centers and widths determined by the nearest 500 patterns. For each helicopter, we created 100 RBF networks, each of which had a different set of centers.[2]

3.3 Ensembles

Ensembles are combinations of multiple *base* models, each of which may be a traditional machine learning model such as an MLP or RBF network. When a new example is to be classified, it is presented to the ensemble's base models and their outputs are combined in some manner (e.g., voting or averaging) to yield the ensemble's prediction. Intuitively, we would like to have base models that perform well and do not make highly-correlated errors. We can see the intuition behind this point graphically in figure 5. The goal of the learning problem depicted in the figure is to separate the positive examples ('+') from the negative examples ('-'). The figure depicts an ensemble of three linear classifiers. For example, classifier C classifies examples above it as negative examples and examples below it as positive examples. Note that none of the three lines separates the positive and negative examples perfectly. For example, classifier C misclassifies all the positive examples in the top half of the figure. Indeed, no linear classifier can separate the positive examples from the negative

[1]That is, for each center, the 300 training cases closest to it in Euclidian distance were used to determine its radius. Therefore, the radius increases with the number of neighboring training cases used.

[2]Due to the large computation time needed to obtain the centers and widths of the kernels on such large data sets, we only used 100 RBF networks as opposed to 280 MLPs.

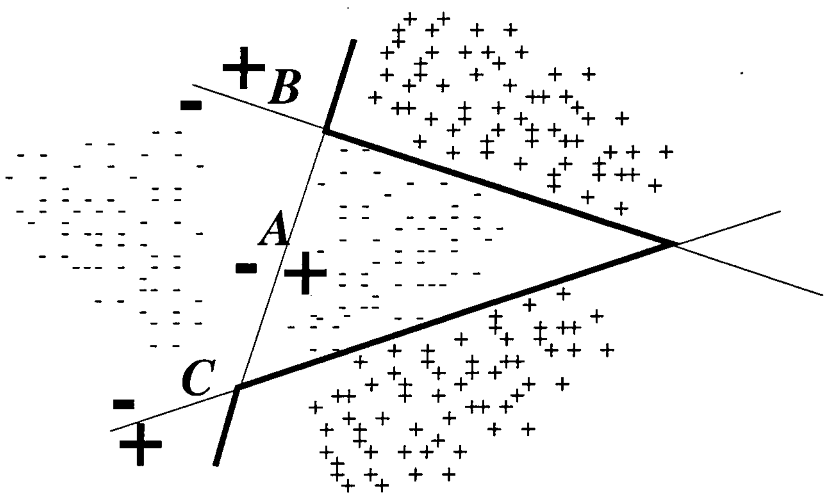

Figure 5: An ensemble of linear classifiers. Each line A, B, and C is a linear classifier. The boldface line is the ensemble that classifies new examples by returning the majority vote of A, B, and C.

examples. However, the ensemble of three lines, where each line gets one vote, correctly classifies all the examples—for every example, at least two of the three linear classifiers correctly classifies it, so the majority is always correct. This is the result of having three very different linear classifiers in the ensemble. This example clearly depicts the need to have base models whose errors are not highly correlated. If all the linear classifiers make mistakes on the same examples (for example if the ensemble consisted of three copies of line A), then a majority vote over the lines would also make mistakes on the same examples, yielding no performance improvement.

The intuition that we have just described has been formalized [14, 16]. Ensemble learning can be justified in terms of the *bias* and *variance* of the learned model. It has been shown that, as the correlations of the errors made by the base models decrease, the variance of the error of the ensemble decreases and is less than the variance of the error of any single base model. If E_{add} is the average additional error of the base models (beyond the Bayes error, which is the minimum possible error that can be obtained), E_{add}^{ave} is the additional error of an ensemble that computes the average of the base models' outputs, and ρ is the average correlation of the errors of the base models, then Tumer and Ghosh [14] have shown that

$$E_{add}^{ave} = \frac{1+\rho(M-1)}{M} E_{add},$$

where M is the number of base models in the ensemble. The effect of the correlations of the errors made by the base models is made clear by this equation. If the base models always agree, then $\rho = 1$; therefore, the errors of the ensemble and the base models would be the same and the ensemble would not yield any improvement. If the base models' errors are independent, then $\rho = 0$, which means the ensemble's error is reduced by a factor of M relative to the base models' errors. It is possible to do even better by having base models with slightly anti-correlated errors. If $\rho = -\frac{1}{M-1}$, then the ensemble's error would be zero.

For both data sets and classifiers, we used simple averaging ensembles. That is, since MLPs and RBF networks return a measure of confidence for each possible class, the averaging ensemble calculates, for each class, the average of all the networks' confidences in that class. The class with the highest average is returned as the ensemble's prediction. Though simple to apply, such ensembles perform remarkably well on a variety of data sets [2, 13, 14]. We experimented with ensembles consisting of 2 to 100 base classifiers for MLP and MLP/RBF ensembles, and 2 to 50 base classifiers for RBF ensembles, although performance improvements after 10 base classifiers were marginal. These ensembles consisted of random samples drawn from the 280 MLPs and 100 RBF networks that we created for the single-network experiments. For each size of ensemble, we drew 20 sets of random samples and report the results as averages over these runs.

3.4 Data Preparation

We created data sets for each of the two aircraft by combining its 176 summary matrices. This resulted in 31475 patterns (revolutions) for the AH1 and 34144 for the OH58c. Both types of classifiers were trained using a randomly-selected two-thirds of the data (21000 examples for the AH1, 23000 for the OH58c) and were tested on the remainder for the first set of experiments. For both aircraft, we used various subsets of the inputs. In particular, for the AH1, we ran experiments using only the bus data and only the vibration data as inputs, in addition to using all the data.

In addition, we calculated the *confusion matrix* of every classifier we created. Entry (i, j) of the confusion matrix of a classifier states the number of times that an example of class i is classified as class j. In examining the confusion matrices of the classifiers (see table 4 for an example of a confusion matrix—entry $(1, 1)$ is in the upper left corner), we noticed that particular maneuvers were continually being confused with one another. In particular, the three hover maneuvers (8-Hover, 9-Hover Turn Left, and 10-Hover Turn Right) were frequently confused with one another and the two coordinated turns (11-Coordinated Turn Left and 12-Coordinated Turn Right) were also frequently confused (the counts associated with these errors are shown in boldface type in table 4.) These sets of maneuvers are similar enough to one another that misclassifications within these groups are unlikely to imply the presence of faults. Therefore, for the second set of experiments, we recalculated the classification accuracies allowing

Table 4: Sample confusion matrix for OH58c (MLP).

True Class	Predicted Class													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	693	0	7	6	79	0	0	0	0	0	0	0	0	0
2	0	679	0	0	0	0	0	0	0	0	0	0	47	0
3	55	1	568	64	31	6	0	11	9	1	11	7	0	3
4	26	0	43	691	15	0	0	3	0	0	0	2	0	1
5	196	0	68	41	412	0	0	0	0	0	2	16	0	0
6	0	0	0	0	0	719	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	1079	0	0	0	0	0	0	0
8	0	9	22	16	0	0	0	**748**	**177**	**97**	11	6	3	0
9	0	1	1	6	0	0	0	**172**	**381**	**162**	4	7	6	0
10	0	4	1	6	0	0	0	**186**	**170**	**376**	0	8	13	0
11	4	0	15	4	3	0	0	2	1	0	**494**	**217**	0	0
12	3	0	7	6	4	0	0	2	1	0	**200**	**531**	0	0
13	0	63	0	0	0	0	0	4	1	0	0	0	712	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	685

for these misclassifications. For our third set of experiments, we consolidated these two sets of maneuvers in the data before running the experiments. That is, we combined the hover maneuvers into one class and the coordinated turns into one class, yielding a total of 11 possible predictions instead of the original 14. We expected the performance to be best for this third set of experiments because, informally, the classifiers do not have to waste resources distinguishing among the two sets of similar maneuvers.

Finally, we used the knowledge that a helicopter needs some time to change maneuvers. That is, two sequentially close patterns are unlikely to come from different maneuvers. To obtain results that use this "prior" knowledge, we tested on sequences of revolutions by averaging the classifiers' outputs on a window of examples surrounding the current one. In one set of experiments, we averaged over windows of size 17 (8 revolutions before the current one, the current one, and 8 revolutions after the current one) which corresponds to about three seconds. Because the initial training and test sets were randomly chosen from this sequence, this averaging could not be performed on the test set alone. Instead it was performed on the full data set for both helicopters. To allow meaningful comparisons of these results, we also computed the errors of the single-revolution classifiers on this full dataset and present them in tables 6 and 8.[3]

4 Results

In this section we describe the experimental results that we have obtained so far. We first discuss results on the OH58c helicopter. In table 5, the column

[3]We performed this windowed averaging as though the entire data were collected over a single flight. However, it was in fact collected in stages, meaning that there are no transitions between maneuvers. We show these results to demonstrate the applicability of this method to sequential data obtained in actual flight after training the network on "static" single revolution patterns.

Table 5: OH58c Single Revolution Test Set Results.

Base Type	N	Single Rev	Corr	Post-Run Consolidated	Corr	Pre-Run Consolidated	Corr
MLP	1	79.789 ± 0.072	–	92.709 ± 0.055	–	93.566 ± 0.060	–
	4	81.997 ± 0.065	0.4193	93.820 ± 0.044	0.4118	94.422 ± 0.038	0.4443
	10	82.441 ± 0.045	0.4193	94.015 ± 0.028	0.4133	94.672 ± 0.032	0.4395
	100	82.771 ± 0.016	0.4199	94.133 ± 0.011	0.4139	94.672 ± 0.032	0.4374
RBF	1	75.451 ± 0.103	–	89.305 ± 0.080	–	90.460 ± 0.169	–
	4	75.817 ± 0.048	0.7164	89.485 ± 0.047	0.7046	90.912 ± 0.056	0.5877
	10	75.871 ± 0.040	0.7185	89.498 ± 0.034	0.7058	90.987 ± 0.032	0.6009
	50	75.908 ± 0.016	0.7162	89.506 ± 0.011	0.7058	91.018 ± 0.014	0.6028
MLP/ RBF	2	80.190 ± 0.079	0.3687	92.834 ± 0.065	0.3176	93.777 ± 0.046	0.2905
	4	80.946 ± 0.059	0.4352	93.189 ± 0.042	0.3997	94.097 ± 0.048	0.3788
	10	81.406 ± 0.043	0.4574	93.403 ± 0.039	0.4273	94.348 ± 0.025	0.3941
	100	81.543 ± 0.020	0.4681	93.463 ± 0.017	0.4392	94.457 ± 0.011	0.4056

marked "Single Rev" shows the results of running individual networks and ensembles of various sizes on the summary matrices randomly split into training and test sets. We only present results for some of the ensembles we constructed due to space limitations and because the ensembles exhibited relatively small gains beyond $N = 10$ base models. MLPs and ensembles of MLPs outperform RBFs and ensembles of RBFs consistently. The ensembles of MLPs improve upon single MLPs to a greater extent than ensembles of RBF networks do upon single RBF networks, indicating that the MLPs are more diverse than the RBF networks. This is corroborated by the fourth column (marked "Corr")[4] which shows that the average correlations among the base models are much higher for ensembles of RBF networks than ensembles of MLPs. Mixed ensembles perform worse than pure-MLP ensembles and better than pure-RBF ensembles for all numbers of base models. The correlations of the mixed ensembles are larger than those of the pure-MLP ensembles. This shows that RBF networks did not add enough diversity to make mixed ensembles outperform pure-MLP ensembles. The standard errors of the mean performances decrease with increasing numbers of base models as is normally the case with ensembles. The column marked "Post-Run Consolidated" shows the single revolution results after allowing for confusions among the hover maneuvers and among the coordinated turns, consolidating them into single classes (hover and coordinated turns). As expected, the performances improved dramatically. The column "Pre-Run Con-

[4]Each correlation in this paper is the average of the correlations of every pair of base classifiers in the ensemble. We calculate the correlation of a pair of classifiers as the number of test patterns that the two classifiers agree on but misclassify, divided by the number of patterns that at least one classifier misclassifies. Note that this is not the posterior-based correlation used in [14, 15]. A correlation of "x" in any table indicates that there were no training patterns misclassified by any base classifiers; therefore, the correlation is undefined.

Table 6: OH58c Full Data Set Results.

Base Type	N	Window of 17	Corr	Window 17 Post-Consolidated	Corr	Window 17 Pre-Consolidated	Corr
MLP	1	89.905 ± 0.121	–	96.579 ± 0.066	–	97.586 ± 0.078	–
	4	90.922 ± 0.074	0.5014	96.799 ± 0.026	0.6145	97.635 ± 0.041	0.6258
	10	91.128 ± 0.064	0.5013	96.820 ± 0.018	0.6255	97.729 ± 0.031	0.6067
	100	91.307 ± 0.015	0.5052	97.063 ± 0.140	0.6290	97.695 ± 0.006	0.6086
RBF	1	82.564 ± 0.154	–	92.831 ± 0.103	–	94.611 ± 0.124	–
	4	82.634 ± 0.059	0.7509	92.882 ± 0.047	0.7755	94.548 ± 0.063	0.5870
	10	82.618 ± 0.055	0.7543	92.895 ± 0.043	0.7758	94.517 ± 0.029	0.6001
	50	82.644 ± 0.019	0.7505	92.901 ± 0.013	0.7747	94.524 ± 0.012	0.6072
MLP/ RBF	2	88.674 ± 0.108	0.3652	95.910 ± 0.059	0.3596	97.155 ± 0.045	0.3419
	4	88.895 ± 0.078	0.4520	95.902 ± 0.040	0.4791	97.145 ± 0.067	0.4383
	10	89.140 ± 0.057	0.4788	95.980 ± 0.033	0.5143	97.226 ± 0.032	0.4576
	100	89.320 ± 0.025	0.4937	96.003 ± 0.012	0.5335	97.204 ± 0.009	0.4706

Base Type	N	Single Rev	Corr	Single Rev Post Consolidated	Corr	Single Rev Pre-Consolidated	Corr
MLP	1	82.097 ± 0.072	–	93.539 ± 0.058	–	94.495 ± 0.064	–
	4	84.304 ± 0.049	0.4069	94.622 ± 0.039	0.4019	95.321 ± 0.035	0.4443
	10	84.750 ± 0.043	0.4075	94.805 ± 0.028	0.4029	95.540 ± 0.029	0.4372
	100	85.048 ± 0.012	0.4081	94.922 ± 0.011	0.4036	95.595 ± 0.008	0.4355
RBF	1	76.406 ± 0.099	–	89.680 ± 0.077	–	90.788 ± 0.147	–
	4	76.799 ± 0.040	0.7164	89.872 ± 0.039	0.7142	91.187 ± 0.045	0.6027
	10	76.836 ± 0.033	0.7186	89.902 ± 0.027	0.7162	91.244 ± 0.027	0.6157
	50	76.910 ± 0.011	0.7162	89.948 ± 0.007	0.7143	91.271 ± 0.013	0.6182
MLP/ RBF	2	82.146 ± 0.075	0.3613	93.523 ± 0.061	0.3172	94.587 ± 0.049	0.2883
	4	82.877 ± 0.053	0.4293	93.854 ± 0.041	0.4022	94.876 ± 0.051	0.3783
	10	83.332 ± 0.036	0.4516	94.066 ± 0.029	0.4291	95.089 ± 0.024	0.3948
	100	83.505 ± 0.015	0.4618	94.142 ± 0.015	0.4406	95.163 ± 0.014	0.4076

solidated" shows the single revolution results on the summary matrices in which the hovers and coordinated turns were consolidated as described in section 3.4. The performances here were consistently the highest as we hypothesized.

The top half of table 6 shows the results of performing the windowed averaging described in the previous section in the column marked "Window of 17." The columns "Window 17 Post-Consolidated" and "Window 17 Pre-Consolidated" give the results allowing for the confusions mentioned earlier. The bottom half of the table gives the full set errors of the single-revolution classifiers. We can clearly see the benefits of windowed averaging, which serves to smooth out some of the noise in the data.

Table 7 shows the results with the AH1 summary matrices randomly split into training and test sets. Table 8 has the windowed averaging and single-revolution classifier results, respectively, on the full AH1 dataset. These results are substantially better than the OH58c results. We expected this because

Table 7: AH1 Single Revolution Test Set Results.

Base Type	N	Single Rev	Corr	Post-Run Consolidated	Corr	Pre-Run Consolidated	Corr
MLP	1	96.752 ± 0.059	–	99.843 ± 0.032	–	99.990 ± 0.002	–
	4	97.284 ± 0.031	0.4155	99.975 ± 0.010	0.1100	99.997 ± 0.001	x
	10	97.448 ± 0.027	0.4130	99.992 ± 0.001	x	99.994 ± 0.001	x
	100	97.542 ± 0.006	0.4128	99.995 ± 0.001	x	99.992 ± 0.001	x
RBF	1	95.669 ± 0.059	–	99.626 ± 0.017	–	99.695 ± 0.011	–
	4	95.946 ± 0.029	0.6462	99.706 ± 0.010	0.3750	99.751 ± 0.009	0.4230
	10	95.911 ± 0.023	0.6561	99.711 ± 0.006	0.3824	99.757 ± 0.005	0.4251
	50	95.946 ± 0.009	0.6538	99.716 ± 0.003	0.3791	99.761 ± 0.002	0.4215
MLP/ RBF	2	97.040 ± 0.054	0.3120	99.980 ± 0.004	0.0045	99.994 ± 0.002	0.0049
	4	97.318 ± 0.025	0.3698	99.986 ± 0.003	0.0859	99.998 ± 0.001	x
	10	97.429 ± 0.018	0.4040	99.990 ± 0.002	0.1222	99.998 ± 0.001	x
	100	97.521 ± 0.011	0.4160	99.998 ± 0.001	x	100.000 ± 0.000	x

the AH1 is a heavier helicopter, so it is less affected by conditions that tend to introduce noise such as wind changes. With the AH1's summary matrices without consolidation, the mixed ensembles outperform the pure ensembles for small numbers of base models but perform worse than the MLP ensembles for larger numbers of base models. With consolidation, the mixed ensembles outperform the pure ensembles more often; however, the performances are all very high. Once again, we can see that ensembles of MLPs outperform single MLPs to a greater extent than ensembles of RBF networks outperform single RBF networks, so the RBFs are not as different from one another. Unlike with the OH58c, with the AH1, adding a few RBF networks to an MLP ensemble helped. The standard errors of the mean performances tend to decrease with increasing numbers of base models just as with the OH58c.

On the AH1, the hover maneuvers were frequently confused just as they were on the OH58c, but the coordinated turns were not confused. Taking this confusion into account boosted performance significantly. The windowed averaging approach did not always yield improvement when allowing for the maneuver confusions, but helped when classifying across the full set of maneuvers. However, in all cases when windowed averaging did not help, the classifier performance was at least 99.6%, so there was very little room for improvement.

5 Discussion

In this paper, we presented an approach to fault detection that contains a subsystem to classify an operating aircraft into one of several states. More specifically, the proposed subsystem determines the maneuver being performed by an

Table 8: AH1 Full Data Set Results.

Base Type	N	Window of 17	Corr	Window 17 Post-Consolidated	Corr	Window 17 Pre-Consolidated	Corr
MLP	1	98.344 ± 0.059	–	99.737 ± 0.028	–	100.000 ± 0.000	–
	4	98.757 ± 0.031	0.4052	99.811 ± 0.005	0.4111	100.000 ± 0.000	x
	10	98.779 ± 0.021	0.4105	99.815 ± 0.002	0.4044	100.000 ± 0.000	x
	100	98.861 ± 0.006	0.4055	99.816 ± 0.001	0.4127	100.000 ± 0.000	x
RBF	1	96.662 ± 0.102	–	99.404 ± 0.013	–	99.653 ± 0.010	–
	4	96.988 ± 0.042	0.6668	99.431 ± 0.012	0.6607	99.659 ± 0.021	0.6098
	10	96.968 ± 0.028	0.6764	99.428 ± 0.008	0.6636	99.676 ± 0.007	0.6049
	50	97.003 ± 0.008	0.6735	99.438 ± 0.003	0.6645	99.696 ± 0.003	0.6076
MLP/ RBF	2	98.256 ± 0.064	0.2313	99.690 ± 0.006	0.1915	99.908 ± 0.003	0.0000
	4	98.482 ± 0.034	0.3148	99.682 ± 0.004	0.3085	99.901 ± 0.013	x
	10	98.475 ± 0.028	0.3577	99.683 ± 0.003	0.3396	99.918 ± 0.002	x
	100	98.553 ± 0.005	0.3739	99.687 ± 0.001		99.920 ± 0.001	x
Base Type	**N**	**Single Rev**	**Corr**	**Single Rev Post-Consolidated**	**Corr**	**Single Rev Pre-Consolidated**	**Corr**
MLP	1	96.933 ± 0.060	–	99.826 ± 0.037	–	99.992 ± 0.009	–
	4	97.555 ± 0.025	0.3966	99.975 ± 0.014	0.3795	99.997 ± 0.007	x
	10	97.683 ± 0.013	0.3973	99.994 ± 0.009	0.3824	99.997 ± 0.005	x
	100	97.762 ± 0.008	0.3981	99.996 ± 0.009	0.3836	99.997 ± 0.001	x
RBF	1	95.743 ± 0.067	–	99.676 ± 0.014	–	99.726 ± 0.012	–
	4	96.063 ± 0.032	0.6369	99.738 ± 0.005	0.6192	99.767 ± 0.008	0.4129
	10	96.042 ± 0.026	0.6456	99.742 ± 0.009	0.6272	99.773 ± 0.009	0.4156
	50	96.067 ± 0.005	0.6321	99.747 ± 0.000	0.6137	99.781 ± 0.002	0.4150
MLP/ RBF	2	97.231 ± 0.055	0.2933	99.984 ± 0.000	0.0073	99.997 ± 0.005	0.0025
	4	97.502 ± 0.028	0.3539	99.988 ± 0.005	0.0915	99.998 ± 0.005	x
	10	97.570 ± 0.018	0.3899	99.993 ± 0.005	0.1225	99.999 ± 0.003	x
	100	97.659 ± 0.008	0.3978	99.999 ± 0.005	x	100.000 ± 0.000	x

Table 9: AH1 Bus and Non-Bus Results

Inputs	Single Rev	Single Rev Consolidated	Window of 17	Window of 17 Consolidated
All	96.752 ± 0.059	99.843 ± 0.032	98.344 ± 0.059	99.737 ± 0.028
Bus	90.380 ± 0.110	95.871 ± 0.091	91.209 ± 0.126	96.027 ± 0.086
Non-Bus	87.884 ± 0.228	93.731 ± 0.171	92.913 ± 0.355	96.110 ± 0.236
$P(agree)$	79.523 ± 0.247	90.063 ± 0.202	85.609 ± 0.320	93.393 ± 0.247

aircraft as a function of vibration data and any other available data. Through experiments with two helicopters, we demonstrated that the subsystem is able to determine the maneuver being performed with good reliability. These results show great promise in classifying the correct maneuver with high certainty. Future work will involve applying this approach to "free-flight data", where the maneuvers are not static or steady-state, and transitions between maneuvers exist.

The results presented in this paper address the maneuver classification portion of the online fault detection system envisioned in this research and shown in figure 1. To address the overall detection problem, future work will involve experiments to determine the probabilities of agreement between different classifiers, to detect possible faults when there is a mismatch. For example, for the AH1 helicopter, just the data from a 1553 bus (as described in section 2) were used to train some classifiers and compared to other classifiers that used all except the bus data. Table 9 shows the results of training 20 single MLPs on these data using the same network topology as for the other MLPs trained on all the AH1 data. They performed much worse than the single MLPs trained with all the inputs presented at once. The last line in the table indicates the percentage of maneuvers for which the two types of classifiers agreed.

Recall from section 1 that we would like classifier disagreement to indicate the presence of a fault; therefore, we would like these agreement probabilities to be much higher. However, we hypothesize that we can use the bus data in a much simpler way. For example, if a vibration data-based classifier predicts that the aircraft is performing a forward flight, but the bus data indicate that altitude is zero, then the probability of a fault is high. We do not necessarily need a system that returns the maneuver as a function of all the variables that constitute the bus data. In this example, we merely need to know that a zero altitude is inconsistent with a forward flight. We plan to perform a detailed study of the collected bus data so that we may construct simple classifiers representing knowledge of the type just mentioned and use them to find inconsistencies such as what we just described.

There is ongoing work within our research group to model aircraft engine operation from "first principles." In particular, models of the gear system are being prepared so that simulated data may be collected. We plan to use this simulation to insert cracks and other types of faults in the gear system in order to learn how the data changes as a function of these faults. This information can be used to mathematically insert faults into the real data. This gives us the fault data that we clearly cannot collect from the aircraft directly. We hope to generate such fault data and test whether our classification subsystems react to fault data in the way we expect.

References

[1] C. M. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, New York, 1995.

[2] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[3] A.E. Bryson and Y.-C. Ho. *Applied Optimal Control.* Blaisdell Publishing Co., 1969.

[4] Robert Campbell, Amulya Garga, Kathy McClintic, Mitchell Lebold, and Carl Byington. Pattern recognition for fault classification with helicopter vibration signals. In *American Helicopter Society 57th Annual Forum*, 2001.

[5] Paul Hayton, Bernhard Schölkopf, Linel Tarassenko, and Paul Anusiz. Support vector novelty detection applied to jet engine vibration spectra. In Todd K. Leen, Thomas G. Dietterich, and Volker Tresp, editors, *Advances in Neural Information Processing Systems-13*, pages 946–952. Morgan Kaufmann, 2001.

[6] Edward M. Huff, Irem Y. Tumer, Eric Barszcz, Mark Dzwonczyk, and James McNames. Analysis of maneuvering effects on transmission vibration patterns in an AH-1 cobra helicopter. *Journal of the American Helicopter Society*, 2002.

[7] Nathalie Japkowicz, Catherine Myers, and Mark Gluck. A novelty detection approach to classification. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 518–523, Montreal, Canada, 1995. Morgan Kaufmann.

[8] M. A. Kramer and J. A. Leonard. Diagnosis using backpropagation neural networks–analysis and criticism. *Computers and Chemical Engineering*, 14(12):1323–1338, 1990.

[9] D.A. McAdams and I.Y. Tumer. Towards failure modeling in complex dynamic systems: impact of design and manufacturing variations. In *ASME Design for Manufacturing Conference*, volume DETC2002/DFM-34161, September 2002.

[10] Sunil Menon and Rida Hamza. Machine learning methods for helicopter hums. In *Proceedings of the 56th Meeting of the Society for Machinery Failure Prevention Technology*, pages 49–55, 2002.

[11] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition.* Bradford Books/MIT Press, Cambridge, Mass, 1986.

[12] I.Y. Tumer and E.M. Huff. On the effects of production and maintenance variations on machinery performance. *Journal of Quality in Maintenance Engineering*, 8(3):226–238, 2002.

[13] K. Tumer and J. Ghosh. Analysis of decision boundaries in linearly combined neural classifiers. *Pattern Recognition*, 29(2):341–348, February 1996.

[14] K. Tumer and J. Ghosh. Error correlation and error reduction in ensemble classifiers. *Connection Science, Special Issue on Combining Artificial Neural Networks: Ensemble Approaches*, 8(3 & 4):385–404, 1996.

[15] K. Tumer and J. Ghosh. Linear and order statistics combiners for pattern classification. In A. J. C. Sharkey, editor, *Combining Artificial Neural Nets: Ensemble and Modular Multi-Net Systems*, pages 127–162. Springer-Verlag, London, 1999.

[16] Kagan Tumer. *Linear and Order Statistics Combiners for Reliable Pattern Classification.* PhD thesis, The University of Texas, Austin, TX, May 1996.

[17] V. Venkatasubramanian, R. Vaidyanathan, and Y. Yamamoto. Process fault detection and diagnosis using neural networks—i. steady-state processes. *Computers and Chemical Engineering*, 14(7):699–712, 1990.

[18] D. H. Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.